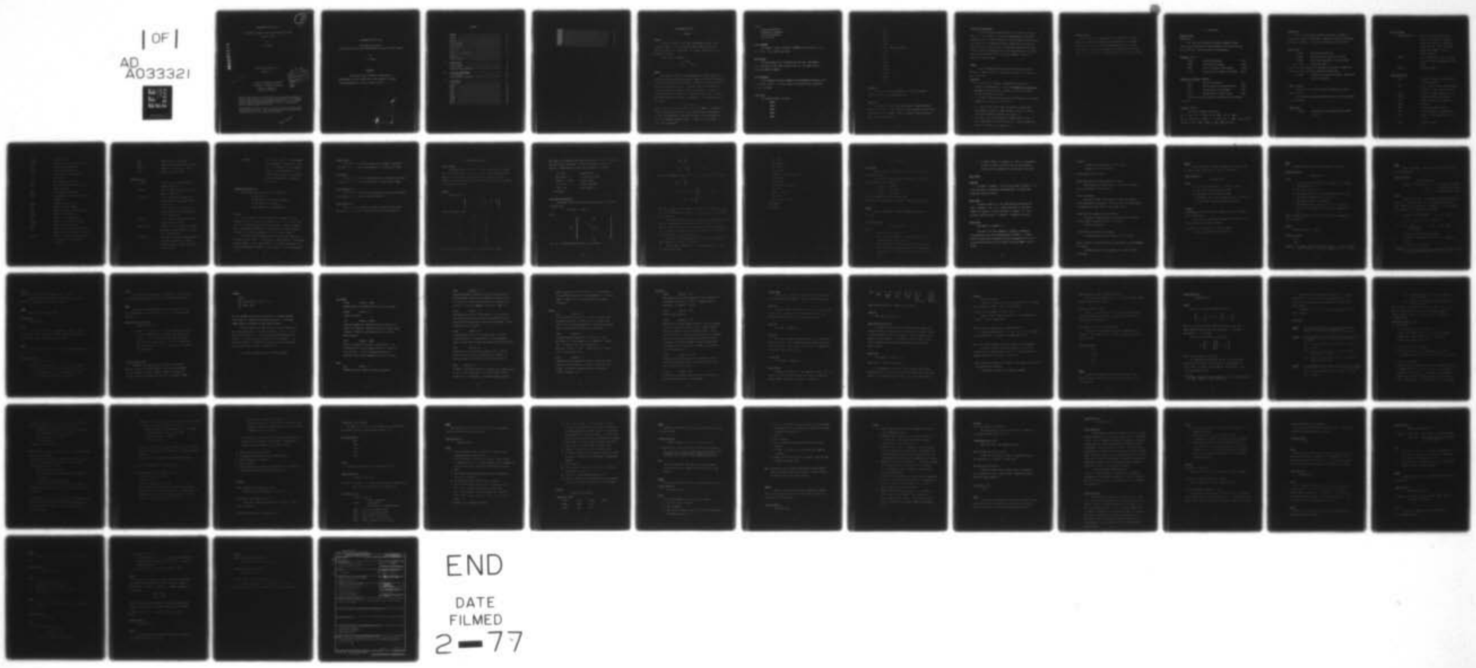

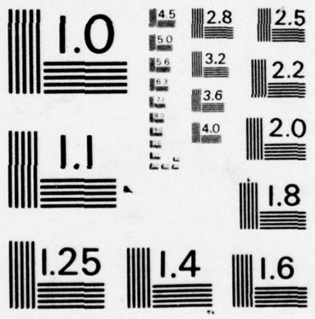
1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(1)

B. S.

ADA033321

PROGRAMMERS GUIDE TO LCPL

A Program for Solving Linear Complementarity Problems

by Lemke's Method

by

J. A. Tomlin

TECHNICAL REPORT SOL 76-25

October 1976

See 1473

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
Stanford University
Stanford, California

Research and reproduction of this report were partially supported by the U.S. Army Research Office-Durham, DAAC-29-74-C-0034; U.S. Energy Research and Development Administration Contract E(04-3)-326 PA #18; Office of Naval Research Contract N00014-75-C-0865; and National Science Foundation Grant DCR75-04544.

408765

PROGRAMMERS GUIDE TO LCPL

Documentation for LCPL--
A Program for Solving Linear Complementarity Problems by Lemke's Method

by

J. A. Tomlin

Abstract

This report gives programmers information and documentation for LCPL--an efficient robust program for solving Linear Complementarity Problems by Lemke's Method.

CONTENTS

DOCUMENTATION OF LCPL

I. GENERAL

Purpose

LCPL is a program to solve linear complementary problems (LCPs) by Lemke's method [2]. It is designed to handle problems with a few hundred rows and a sparse coefficient (M) matrix. Formally an LCP is a problem of the form:

Find w and z such that

$$w = q + Mz$$

$$w, z \geq 0, \qquad w^T z = 0 .$$

Method

The code uses a product form implementation of Lemke's method, with an efficient inversion routine producing a sparse and accurate LU decomposition of the basis in product form. The basis is updated by the standard product form method with accuracy checks. The Harwell scaling subroutine MC12A using Curtis and Reid's scaling method is incorporated. A routine for recovering from loss of feasibility or singularity of the basis is provided. The methodology is fully described in reference [4]. The problem is stored in the form

$$Iw - Mz - e\zeta = q$$

where e has a +1 on every row with $q_i < 0$. ζ ("DUMMY Z") is immediately introduced into the basis to give a basic feasible almost complementary solution. The complement of the ejected variable is then introduced and so on. The algorithm terminates when $\zeta <$ ZTOLZE (or ζ is non-basic) or when a ray is encountered.

Keywords

Linear Complementarity

Mathematical Programming

Quadratic Programming

Sparse Matrices

Source Language

The program is written entirely in FORTRAN IV for IBM series 360 and 370 computers. It is WATFIV compatible.

Specification

The main program should be executed as a job step. The program input is described in the "Users Guide to LCPL" [3]. All input occurs in subroutines INPUT and BASINP.

Error Indicators

Error indicators and other diagnostics and messages are described in the "Users Guide to LCPL" [3]. They will also be indicated in the documentation for each subroutine.

Subroutines

The routines making up LCPL are

BASINP

BASOUT

CHSOL

CHUZR

CLEAR

FTRAN

ITEROP

INPUT

INVERT

LEMKE

MC12A (Harwell Subroutine)

OUTPUT

RECOVR

SCALE

SHFTE

SHIFTR

UNPACK

UPBETA

WRETA

XCOLS

<u>Program Size</u>

The total length of the program is 1867 source statements (including comments and common blocks, etc.).

<u>Array Size</u>

The standard version of the code requires a blank common array area of 130,068 bytes of core. This is of course dependent on the setting of maximum problem dimensions. There is a labelled common requiring 80 bytes. This is not variable.

Accuracy and Convergence

The code has been specially written to be as robust as possible (see [4]). If the matrix M is positive semi-definite or has positive principal minors, a recovery procedure is provided (RECOVR) which virtually assures that loss of feasibility or basis singularity can be overcome. This facility also allows the user to start such problems from any complementary or almost complementary basis. Note that convergence is assumed when the dummy variable "DUMMY Z" falls below ZTOLZE. It need not necessarily leave the basis.

Timing

Using the FORTRAN H compiler with OPT = 2 the code has solved problems of order 100 in about 4 seconds and of order 500 in 20 seconds on an IBM 370/168.

References

[1] R.W. Cottle and G.B. Dantzig, "Complementary Pivot Theory of Mathematical Programming," pp. 115-138 in Mathematics of the Decision Sciences (G.B. Dantzig and A.F. Veinott, Jr., Eds.), Amer. Math. Society, Providence, R.I. (1968).

[2] C. E. Lemke, "Bimatrix Equilibrium Points and Mathematical Programming," Management Science 11, pp. 681-689 (1965).

[3] J.A. Tomlin, "Users Guide to LCPL--A Program for Solving Linear Complementarity Problems by Lemke's Method," TR SOL 76-16, Systems Optimization Laboratory, Stanford, Ca., August 1976.

[4] J.A. Tomlin, "Robust Implementation of Lemke's Method for the Linear Complementarity Problem." TR SOL 76-24, Systems Optimization Laboratory, Stanford, Ca., September 1976.

Further Comments

Since virtually all communication is via common blocks, and all arrays are in common, the contexts of both labelled and blank common are described in the next section. Furthermore since the code uses sparse matrix techniques for both matrix and transformations, and these appear in many subroutines, these techniques are also described in detail.

II. COMMON BLOCKS

Labelled Common

COMMON/BLOCK/

This common block contains tolerances, parameters giving maximum array dimensions and literal constants. All these values are set in BLOCK DATA. Standard values are in parentheses.

Tolerances (REAL*4)

ZTOLZE	Feasibility tolerance	(1.E-4)
ZTOLPV	Absolute pivot tolerance	(1.E-6)
ZTETA	Absolute tolerance on eta elements	(1.E-12)
ZTOLRP	Relative pivot tolerance used in CHUZR	(1.E-8)

Dimensional Parameters (INTEGER*4)

NRMAX	Maximum row dimension	(350)
NTMAX	Maximum number of eta transformations	(2000)
NEMAX	Maximum number of eta elements	(8000)
NAMAX	Maximum number of matrix nonzeros	(4000)

These values must be changed if the relevant array dimensions are changed.

Literals (INTEGER*4)

All literal constants begin with Q.

QA is 'Abbb', QC is 'Cbbb', QE is 'Ebbb', QH is 'Hbbb', QL is 'Lbbb', QN is 'Nbbb', QO is 'Obbb', QR is 'Rbbb', QU is 'Ubbb' and QBL is 'bbbb' (blank), QDUM1 is 'DUMM', QDUM2 is 'YbZb'.

Blank Common

Almost all communication between subroutines is by means of variables and arrays in blank common. (The major exception is the Harwell scaling routine MC12A.) We indicate the dimension with each array.

REAL*8 Arrays

B (NRMAX)	The given right hand side
X (NRMAX)	Holds the values of the basic variables
Y (NRMAX)	Work region (used mainly to update columns)
YTEMP (NRMAX)	Second work region

The 4 arrays are contiguous and considered to be arrays 1 through 4 by subroutine SHIFTR which moves information between them.

E (NEMAX)	Contains the non-zero eta elements. (They are not divided by the pivot.)

REAL*8 Variables

DSUM, DPROD, DY, DE and DP are used for temporary storage in several routines.

ERMAX contains the maximum relative error found in the right hand side during CHSOL.

REAL*4 Array

A (NAMAX)	Contains the non-zero elements of the matrix (I,-M,-e)

INTEGER*4 Arrays

ICNAM (2*NRMAX+2,2)	Contains the row and column names. ICNAM(J,1) and ICNAM(J,2) contain an 8 character row name for J $\leq$ NRØW and a column name for J > NROW. The "dummy column" name "DUMMY Z" is in the 2*NROW+1 positions.
NAME (20)	Used in reading input cards. The first two words contain the problem name after INPUT.
NTEMP (20)	Temporary storage.

INTEGER*4 Variables

IBFRQ	Iteration frequency for saving the basis (if KOUTB specified)
IFNEG	Switch to reverse sign of matrix elements. Standard is 1 (yes). Set to 0 for no.
IFSCAL	Switch for scaling. Standard is 0 (no).
INVFRQ	Iteration frequency for inversion
IOUT	Output option switch. Standard is 0.
IRØWP	Pivot row
IRØWZ	Row on which "DUMMY Z" pivots
ITCH	Iteration frequency for calling CHSOL
ITCHA	Counter of iterations since last call to CHSOL
ITCNT	Iteration counter

ITRLIM	Iteration limit
ITSINV	Count of iterations since last INVERT
IVIN	Vector number entering basis
IVOUT	Vector number leaving basis
JCOLP	Vector (column) to be updated and enter basis
KINB	Unit number for basis input
KINP	Unit number for problem input in MPS format (Standard 5)
KØUTB	Unit number for saving bases
KOUTP	Unit number for printing final solution (Standard 6)
MSTAT	Problem status (A,C,U or L)
NAEL	Number of A matrix elements
NCOL	Number of columns (should be 2*NROW+1)
NELEM	Number of eta elements
NETA	Number of eta transformations
NLELEM	Number of L eta elements in INVERT
NLETA	Number of L etas in INVERT
NONC	Number of the variable non-basic complementary pair just removed from the basis
NQUAD	Number of quadratic variables (assumed to correspond to the first NQUAD columns of M)

NROW	Number of rows in the problem
NSING	Number of singularities found by INVERT
NUELEM	Number of U eta elements in INVERT
NUETA	Number of U etas in INVERT

INTEGER*2 Arrays

IA (NAMAX)	IA(I) contains the row index for the element stored in A(I).
IE(NEMAX)	IE(I) contains the row index for the eta element stored in E(I)
ISC(NRMAX+1,2)	ISC(I,1) contains an integer, say R, which defines the row scaling factor 16^R for row I of M. ISC(I,2) contains an integer, say C, which defines the column scaling factor 16^C for column I of M.
JH(NRMAX)	JH(I) contains the number of the vector (column) pivoting on row I.
KINBAS(2*NRMAX+2)	KINBAS(J) is zero if column J is nonbasic and equal to the row on which it pivots if the column is basic
LA(2*NRMAX+2)	LA(I) points to the first nonzero element of column I in arrays IA(·) and A(·). The last nonzero of column I is pointed to by LA(I+1)-1.

LE(NTMAX)	LE(I) points to the first nonzero element of transformation I in arrays IE(·) and E(·). This is always the pivot row and element for this transformation. The nonpivot elements follow. The last nonzero of transformation I is pointed to by LE(I+1)-1.

Changing Array Dimensions

The user may change one or all of the following:

Maximum row size

Maximum number of nonzero matrix elements

Maximum number of eta elements

Maximum number of eta transformations.

Row Size

The value NRMAX in labelled common must be changed to the new maximum row size in BLOCK DATA. The arrays B(·), X(·), Y(·), YTEMP(·) and JH(·) must be dimensioned to the new value of NRMAX. Similarly, KINBAS(·) must be dimensioned to 2*NRMAX+2, as should LA(·) and the first dimension of ICNAM(.,2). The first dimension of ISC(.,2) should be NRMAX+1.

In addition, the equivalence statement in subroutine INVERT must be modified so that HREG(1) is equivalenced to YTEMP($\frac{1}{2}$ NRMAX+1). NRMAX should therefore always be even. Finally BARRAY(·) in SHIFTR (equivalenced to B(·)) should be dimensioned to 4*NRMAX, as should WSPACE(·) in SCALE (equivalenced to Y(·)).

Matrix Elements

The value NAMAX in labelled common must be changed in BLOCK DATA. The arrays IA(·), A(·) must be redimensioned to the new value of NAMAX.

Eta Elements

The value of NEMAX in labelled common must be changed in BLOCK DATA. The arrays IE(·), E(·) must be redimensioned to the new value of NEMAX.

Eta Transformations

The value of NTMAX in labelled common must be changed in BLOCK DATA. The dimension of LE(·) should be changed to NTMAX+2.

Overall Array Size

Any of the above changes will change the length of blank common. The array size in subroutine CLEAR should be changed to reflect this.

III. INFORMATION STRUCTURES

Matrix Storage

The problem matrix $(I,-M,-e)$ is stored in the form of a threaded list using the arrays $LA(\cdot)$, $IA(\cdot)$, $A(\cdot)$ which contain respectively: the pointer to the first entry of each column in $IA(\cdot)$ and $A(\cdot)$, the row indexes of the nonzero entries, and the values of the nonzero. All elements of a column are contiguous.

Example.

The small matrix

$$\begin{bmatrix} 1. & & & -4. & & -2 & -1. \\ & 1. & & & -3. & -1 & -1. \\ & & 1. & 2. & 1. & & \end{bmatrix}$$

would be stored as follows

LA	IA	A
1	1	1.
2	2	1.
3	3	1.
4	1	-4.
6	3	2.
8	2	-3.
10	3	1.
12	1	-2.
	2	-1.
	1	-1.
	2	-1.

Note that there is one more entry in $LA(\cdot)$ than there are columns.

This points to the position after the last entry in IA(·) and A(·) enabling this last entry to be located. To unpack say column J into the work region Y(·) (previously set to zero) the code would be as follows

```
        LL = LA(J)                  (Find first and
        KK = LA(J+1) - 1             last entry of column)
        DØ 100 I = LL, KK           (Initiate loop)
        IR = IA(I)                  (Extract row index)
        Y(IR) = A(I)                (Store value in Y(·))
    100 CONTINUE
```

Transformation Processing

The inverse of the basis is maintained as a product of elementary transformations:

$$B^{-1} = E_t E_{t-1} \cdots E_2 E_1$$

where

$$E_k = \begin{bmatrix} 1 & & & & \eta_{1k} & & & & \\ & \cdot & & & \vdots & & & & \\ & & \cdot & & \vdots & & & & \\ & & & \cdot & & & & & \\ & & & & 1 & & & & \\ & & & & \eta_{p_k k} & & & & \\ & & & & \vdots & 1 & & & \\ & & & & \vdots & & \cdot & & \\ & & & & & & & \cdot & \\ & & & & & & & & \cdot \\ & & & & \eta_{mk} & & & & 1 \end{bmatrix}$$

Each η_k is constructed from a vector α_k as follows:

$$\eta_{p_k k} = \frac{1}{\alpha_{p_k k}}$$

$$\eta_{ik} = -\frac{\alpha_{ik}}{\alpha_{p_k k}} \qquad (i \neq k)$$

Hence when multiplying a vector $y^{(k)}$ by E_k to obtain $y^{(k+1)}$ we have

$$y_{p_k}^{(k+1)} = \frac{y_{p_k}^{(k)}}{\alpha_{p_k k}}$$

$$y_i^{(k+1)} = y_i^{(k)} - \alpha_{ik} \cdot \frac{y_{p_k}^{(k)}}{\alpha_{p_k k}}, \qquad (i \neq k)$$

$$= y_i^{(k)} - \alpha_{ik}\, y_{p_k}^{(k+1)}$$

Note that if $y_{p_k}^{(k)}$ is within tolerance of zero, $y^{(k+1)} = y^{(k)}$. Note also that it is only necessary to store the α_{ik} (not the η_{ik}) and the index p_k for each eta.

The nontrival columns for each eta are stored contiguously in the same way as the columns of a matrix using the three arrays LE(·), IE(·) and E(·). Here LE(·) points to the first nonzero of each transformation, IE(·) contain the row indices of the nonzero elements, and E(·) contains the nonzero values of α_{ik} above (not η_{ik}).

The pivot of each transformation is identified by its being the first element in IE(·) and E(·) for that eta.

Thus to apply transformation K to the work vector Y(·) the code could be as follows:

```
      LL = LE(K)
      KK = LE(K+1)-1
      IPIV = IE(LL)
      DY = Y(IPIV)
      IF(DABS(DY) .LT. TØL) GØ TØ 1000
      DY = DY/E(LL)
      Y(IPIV) = DY
      IF (LL.GE. KK) GØ TØ 1000
      LL = LL+1
      DØ 500 I = LL, KK
      IR = IE(I)
      Y(IR) = Y(IR) - E(I)*DY
  500 CONTINUE
 1000 CONTINUE
```

IV. SUBROUTINES

Main Program

Calls the major subroutines of the code--INPUT, LEMKE (the algorithm) and OUTPUT.

The action of the main program is determined by a parameter IACT returned by INPUT as follows:

If IACT < 0 Terminate

If IACT = 0 Call LEMKE

If IACT > 0 Call OUTPUT

OUTPUT is always called after executing LEMKE.

After OUTPUT the program calls INPUT again to look for a new problem.

BASINP

Reads a complementary, or almost complementary basis from file KINB.

Formal Declaration

SUBROUTINE BASINP

Method

0/ Rewinds unit KINB if KINB > 7

1/ Sets JH(·) and KINBAS(·) to specify unit basis and complementary solution. Also NONC = 0

2/ Reads basis name (if any) and checks against problem name.

3/ Reads column names which will replace their complementary unit vectors in the basis (with checking for valid column names).

4/ If column "DUMMY Z" is supplied as a column its accompanying row name is checked for validity and the column replaced the unit vector corresponding to the row name in the basis.

Input Format

NAME Card

Has "NAME" in columns 1 - 4 and the basis name in columns 15 - 22. If this name does not agree with the problem name, or the name card is missing, a warning is given.

Basis Cards

For basic columns of M the column names are given (one per card) in columns 5 - 12. If ζ is basic there must be a card with "DUMMY Z" in columns 5 - 12 and a row name in columns 15 - 22. This row name is the name of the w variable ζ replaces in the basis.

ENDATA Card

Has "ENDATA" in columns 1 - 6.

Note that if the basis supplied is singular or infeasible the program will enter the recovery procedure ("RECOVR"). If naming errors are encountered in reading the basis the program will start with the partial basis found (which will almost certainly trigger a call to RECOVR).

Messages

STARTING BASIS WILL BE READ FROM FILE 'number'.

Message that KOUTB was specified.

WARNING-BASIS NAME CARD MISSING

WARNING-BASIS HAS NAME 'name'-DIFFERENT FROM PROBLEM

Warning messages on reading a basis. May not be the correct basis for the problem. Run continues.

NO MATCH FOR VECTOR

The basis file has a vector name not found in the problem. Vector ignored and run continues. Will almost certainly yield singular or infeasible basis and a call to the recovery routine.

NO MATCH FOR ROW 'rownam' TO SWAP WITH DUMMY Z

The row name 'rownam' cannot be found when reading the basis and the "DUMMY Z" variable is not put in the basis.

The run will continue.

SPURIOUS CARD IN BASIS FILE-'text'-ABORT

An unrecognizable card in the basis file - fatal error.

ERROR IN TRYING TO READ BASIS FROM FILE 'number'-CHECK JCL AND PARAMETERS-RUN ABORTED

A FORTRAN read error has occurred trying to read file KINB. Fatal error.

BASOUT

Writes the names of the basic structural (M) columns, as well as "DUMMY Z" and the row on which it pivots to file KOUTB.

Formal Declaration

SUBROUTINE BASOUT

Method

0/ *If KOUTB not specified exit. If KOUTB > 7 rewind.*

1/ Write NAME card (see BASINP for format)

2/ Write names of basic structural columns

3/ If "DUMMY Z" basic check that NONC defined. If so find unit vector "DUMMY Z" replaces and write their names.

Messages

NONCOMPLEMENTARY PAIR NOT DEFINED, BUT DUMMY Z IN BASIS, UNRESOLVABLE ERROR-NO BASIS SAVED.

The solution is supposedly complementary but DUMMY Z is basic. Highly unlikely unless program corrupted.

BASIS WRITTEN ON FILE 'number' at ITERATION' number'

The basis has been written on file KOUTB.

CHSOL

Checks the relative error in the current basic solution.

Formal Declaration

SUBROUTINE CHSOL

Method

1/ The original right hand side $B(\cdot)$ is shifted to $Y(\cdot)$, updated and then shifted to $X(\cdot)$.

2/ $Y(\cdot)$ is set equal to $B(\cdot)$

3/ The solution vector $X(\cdot)$ is multiplied on the left by the basis and subtracted from $Y(\cdot)$, i.e., $B(\cdot)$.

4/ The residual now in $Y(\cdot)$ is updated.

5/ The maximum absolute values of $Y(\cdot)$ and $X(\cdot)$ are computed and their ratio, ERMAX, gives the estimated relative error in $X(\cdot)$.

ERMAX is tested in INVERT to see if re-inversion should be performed with tighter tolerances.

Message

RELATIVE ERROR IN X = 'number"

Subroutines Called

SHIFTR

FTRAN

Reference: J.A. Tomlin, "Pivoting for Sparsity and Size in Linear Programming Inversion Routines," J. Inst. Maths. Applics. 10 (1972), 289-295.

CHUZR

Selects the variable to leave the basis and the pivot row for the incoming variable.

Formal Declaration

SUBROUTINE CHUZR (IFPV)

Parameter:	IFPV (Output)	set to	-1	if infeasibility observed
		set to	0	if unacceptable pivot found
		set to	1	if acceptable pivot found.

Method

Performs a ratio test on the elements of the updated increasing vector Y(·) and the solution X(·) to determine the pivot row, by P.M.J. Harris' two pass technique.* The solution X(·) is also examined for infeasibility. The technique is as follows:

1/ Find $\mathrm{DPSI} = \min \dfrac{X(I) + \mathrm{ZTOLZE}}{Y(I)}$ for $Y(I) \geq \mathrm{ZTOLPV}$

If $X(I) < -\mathrm{ZTOLZE}$ set IFPV = -1 and exit

2/ Find $\max\limits_{\substack{\mathrm{DTHETA} \leq \mathrm{DPSI} \\ Y(I) \geq \mathrm{ZTOLPV}}} Y(I)$ where $\mathrm{DTHETA} = \dfrac{X(I)}{Y(I)}$

and choose row IROWP which gives maximum.

3/ If $Y(\mathrm{IROWP}) \geq \mathrm{ZTOLRP} * \max|Y(I)|$ set IFPV = 1, otherwise set IFPV = 0.

4/ If $X(\mathrm{IROWP}) < 0$ set to zero.

*Reference: P.M.J. Harris, "Pivot Selection Methods of the Devex LP Code," Mathematical Programming Studies 4, (1975), 30-57.

Message

FEAS LOST ON ROW 'row no." VAR 'variable name' = 'value'

A loss of feasibility is detected when attempting to choose a pivot row. Call to RECOVR is triggered in LEMKE.

CLEAR

Sets all of blank common to zero.

Specification

SUBROUTINE CLEAR

Method

Blank common is defined as a single array - ARRAY(·) - which is set to zero in a loop. The length of ARRAY(·) should be the number of single words in blank common in all other routines.

FTRAN

Updates the work vector Y(·) by the current inverse representation (eta file).

Formal Declaration

SUBROUTINE FTRAN (IPAR)

Parameter: IPAR (Input) is set to 0 if Y(·) is to be updated by the entire eta file (etas 1 through NETA). IPAR is set to 1 to avoid updating Y(·) by above bump etas in INVERT (Y(·) is updated by etas NLETA+1 through NETA.)

Method

The standard LP FTRAN procedure is followed. The data structure of the eta file is described in the section "Transformation Processing."

INPUT

Reads the non-standard parameter settings for the run (if any) and the problem matrix M and right-hand side q in a slightly modified MPS format.

Specification and Parameters

SUBROUTINE INPUT (IACT)

Parameter: IACT (Output) Set to -1 if the subroutine detects the end of the input stream, or if a fatal input error is discovered. Set to 0 if the problem is properly posed and does not have a trivial solution. Set to 1 if problem read has a trivial solution (i.e. unit or given basis is feasible and complementary).

Program Parameter Input

Parameters are changed from their default value for each problem by means of a FORTRAN "NAMELIST" input from the card reader. The first card must start with &PARAM in column 2 or later; each card must begin with a blank, and the list must be terminated by &END.

Examples.

```
cols
1 2 3
    &PARAM NQUAD=64, IFSCAL=1,  END

    &PARAM  &END
```

The first example specifies that the problem is a quadratic program in the first 64 variables and calls for scaling of M. The second example leaves all parameters at their default values.

On normal completion of input we return to the main program with the packed problem matrix [I,-M,-e] stored in LA(·), IA(·), A(·) and the right-hand side in B(·). All the row and column names are held in ICNAM(.,.). The initial basis is defined by JH(·) and KINBAS(·). The row on which "DUMMY Z" has pivoted is stored in IROWZ and the variable thus eliminated from the basis is stored in NONC.

We classify the parameters into the following groups:

Run Control:

ITRLIM (default = 99999)

Maximum number of iterations for this run of this problem.

INVFRQ (default = 30)

Inversion frequency

ITCH (default = INVFRQ)

Iteration frequency for checking the relative error in the current basic solution. Defaulted to do so just before normal inversion. (INVERT automatically carries out this check after inversion.)

IBFRQ (default = 99999)

Iteration frequency for writing the current basis to the file specified by KOUTB (not done unless KOUTB is specified to be non-zero). If KOUTB is properly specified the final basis will be saved regardless of the value of IBFRQ.

Input:

KINP (default = 5)

FORTRAN logical unit number for reading the problem.

IFNEG (default = 1)

Determines whether to change the sign of the input matrix M. With the default setting the user supplies the matrix M and -M is stored to collect all variables on the left-hand side as in (6). If the user <u>supplies</u> -M then set IFNEG = 0.

IFSCAL (default = 0)

This parameter is set to 1 if the matrix M is to be scaled. If scaling is performed the solution (and quadratic objective function value) are descaled to give their true values. Usually scaling will not be required.

IFALLE (default = 0)

If the parameter is set to 1 the dummy column e will be constructed with a 1 in every position. Not recommended for normal use since this increases the density of the transformations.

KINB (default = 0)

FORTRAN logical unit number for reading a basis. With the default value of 0 no basis reading will be attempted. If KINB is greater than 7 the unit is rewound before reading.

NQUAD (default = 0)

If NQUAD is non-zero the problem is assumed to be a quadratic program set up as in the "Users Guide". In this case only the NQUAD columns of M corresponding to the quadratic primal variables need be supplied; the $-A^T$ column will be constructed from the last NROW-NQUAD rows of the column supplied. If M is non-square and NQUAD is not correctly specified a fatal error is generated.

<u>Output:</u>

IØUT (default = 0)

With the default setting all the w and z variables, names and values,are printed. If IØUT is set to a non-zero value only the basic variables are printed (in sorted order) together with the row names and the original right hand side q.

NQUAD (default = 0)

If NQUAD has some non-zero value n the first n rows and columns of M are assumed to be the Hessian of a quadratic form. The value of the quadratic form will be printed out after the w and z variable values at solution time. (See also Input above.)

KOUTB (default = 0)

FORTRAN logical unit number for writing the basis every IBFRQ iterations and on termination. No basis is output if KOUTB is zero. Unit KOUTB is rewound before and after writing if KOUTB is greater than 7.

Tolerances:

ZTOLZE (default = 10^{-4})

Zero tolerance for testing feasibility. Basic variables are non-negative if they are $\geq$ -ZTOLZE. Complementarity is declared if ζ ("DUMMY Z") $\leq$ ZTOLZE.

ZTOLPV (default = 10^{-6})

Absolute pivot tolerance.

ZTOLRP (default = 10^{-8})

Relative pivot tolerance while iterating. A pivot is accepted if its absolute value is greater than the largest absolute value in the updated column multiplied by ZTOLRP. If this test fails INVERT is called and the same variable is considered again. If the chosen pivot element still fails the test it is accepted. Note that if ZTOLRP is large this could lead to inverting at every iteration. If ZTOLRP is too small the test becomes ineffective.

ZTETA (default = 10^{-12})

Tolerance on the absolute value of transformation elements. Larger values may give sparser transformations but at great cost to accuracy and stability.

ZTOLDA (default = 10^{-7})

Tolerance on the absolute value of entries in the matrix M. Any value smaller than this is disregarded.

Problem Input

The M and q defining the problem are read from unit KINP in slightly modified "MPS format." The card images required are:

NAME Card

This has "NAME" in columns 1 to 4 and the (up to 8 characters) problem name in columns 15-22. This card is optional but highly desirable (particularly for checking against basis names).

ROWS Card

Has "ROWS" in columns 1-4.

Row Names

Each row is assigned a (unique) name of up to 8 characters, one per card, in columns 5-12. Embedded blanks are allowed (unlike MPS). Note that there are no row types as in MPS and that any supplied will be ignored. Column 1 must be blank.

COLUMNS Card

Has "COLUMNS" in columns 1-7.

Matrix Element

The non-zero elements of M are supplied by column. All the elements of a column must be together. Each column is assigned a (unique) name of up to eight characters. The format is:

cols	1 - 4	5 - 12	15 - 22	25 - 36	40 - 47	50 - 61
	blank	column name	row name	element value	second row name (optional)	second element value (optional)

Again embedded blanks are allowed in column names.

RHS Card

Has "RHS" in columns 1 - 3.

Right Hand Side Elements

The right hand side vector (q) may be given a name, which should be different from any row or column name. The elements are given in the same format as the matrix elements. Note that only one right hand side may be supplied. If an attempt is made to input more than one, the non-zeros from later right hand sides will over-write the earlier values.

ENDATA Card

Has "ENDATA" in columns 1 - 6.

Sample input is shown in section 4.1.

It is important to note that the "w" variables are assigned the row names of M, and the "z" variables the column names. The dummy column e is assigned the name "DUMMY Z", as is the dummy variable ζ.

Messages

SPLIT VECTOR 'name'

A column with the same 'name' was encountered earlier. The input deck is probably corrupted. Input continues however.

NO MATCH FOR ROW 'rowname' AT COLUMN 'colname'

Column 'colname' contains a nonzero on an unidentifiable row 'rowname'. This is a fatal error.

NUMBER OF MATRIX ELEMENTS 'number' EXCEEDS NAMAX-ABORT

The program dimensions for $A(\cdot)$ and $IA(\cdot)$ must be increased to solve the problem. Fatal error. Standard NAMAX is 4000.

MATRIX NOT SQUARE AND NQUAD INCORRECTLY SPECIFIED-PROBLEM ABANDONED.

The number of structural columns supplied is not equal to either NROW or the specified value of NQUAD. M cannot therefore be properly constructed to be square. Fatal error.

NQUAD COLUMNS ONLY SUPPLIED--WILL ATTEMPT TO FORM 'number' EXTRA COLUMNS OF CONSTRAINT MATRIX TRANSPOSE

The problem is assumed to be a quadratic program

DUMMY COLUMN WITH 'number' ELEMENTS CONSTRUCTED

There are negative elements in the q vector and a dummy column has been constructed for "DUMMY Z".

PROBLEM HAS TRIVIAL SOLUTION

Either q is non-negative or the given starting basis yields a feasible complementary solution. No dummy column is constructed and the solution is printed out.

BASIS GIVEN NOT FEASIBLE, WILL ENTER RECOVR

The given basis (complementary or almost complementary) is infeasible. The recovery routine is called to construct a new basis and dummy column.

<u>Subroutines Called</u>

CLEAR

BASINP

RECOVR

INVERT

XCOLS

INVERT

Performs an LU decomposition of the basis, using a "Merit Method" to preserve sparsity, and writes the product form of $U^{-1}L^{-1}$ to the eta file.

Formal Declaration

SUBROUTINE INVERT

Method[*]

The basis is (notionally) permuted to the form

$$\begin{bmatrix} L & & \\ A & B & \\ C & E & L_2 \end{bmatrix} \quad \text{and then} \quad \begin{bmatrix} L_1 & & \\ \bar{C} & U_1 & \bar{E} \\ A & & B \end{bmatrix}$$

where L_1 is lower triangular (the above the bump columns) and U_1 upper triangular, permuted from lower triangular L_2 (the below bump columns). B is the bump.

The resulting factorization is of the form

$$LU = \begin{bmatrix} L_1 & & \\ \bar{C} & I & \\ A & & \bar{L}_2 \end{bmatrix} \begin{bmatrix} I & & \\ & U_1 & \bar{E} \\ & & \bar{U}_2 \end{bmatrix}$$

where B has triangular factors $\bar{L}_2\bar{U}_2$.

The routine proceeds by pulling out all the above and below bump etas and writing them out, then writing the bump columns in "merit" order. The bump is then decomposed. The resulting L and U etas are merged by SHFTE.

[*] J. A. Tomlin, "Pivoting for Size and Sparsity in L.P. Inversion Routines," J. Inst. Maths. Applics. 10, pp. 289-295 (1972).

On completion the right-hand side is placed in Y(·), updated and put in X(·). CHSOL is called to check relative error in X(·). if this is too large reinversion is performed with a larger relative pivot tolerance.

Since INVERT is the most intricate routine we describe it more fully as follows:

Local Arrays

H-region (HREG)	Contains sequence numbers of vectors already pivoted on. Contains -1 - Row Count for unassigned rows Contains 0 for rows below the bump and not yet pivoted in.
V-region (VREG)	Contains sequence numbers of columns not yet pivoted into rows. It is divided into 4 parts (some of which may be vacuous) as follows: Part 1 contains sequence numbers of columns in the bump. Part 2 is spare. Part 3 contains sequence numbers of structural columns pivoting below the bump Part 4 contains sequence numbers of slacks and artificials pivoting below the bump
M-region (MREG)	Contains information about those rows mentioned in the V-region. It is divided into 4 parts exactly like the V-region as follows: Part 1 contains the merit measure of columns in the bump Part 2 is spare.

Part 3 contains the pivot rows of vectors mentioned in Part 3 of V-region.

Part 4 contains the pivot rows of vectors mentioned in Part 4 of V-region

Note that MREG(·) is equivalenced to YTEMP(·), VREG(·) to X(·) and HREG(·) to YTEMP($\frac{1}{2}$ NRMAX+1). Thus the last equivalence must be changed if NRMAX is changed. The three arrays are INTEGER*2 and of dimension NRMAX.

OUTLINE FLOW CHART FOR INVERT

1. Scan H-region

   If a row contains a slack or artificial,

      put sequence number into Part 4 of M-region and V-region

   Otherwise put sequence number into Part 1 of V-region.

   In either case replace H-region by -1.

2. Take vectors in Part 4 of M-region

   Zero out those entries in the H-region mentioned in Part 4

      of M-region, and pivot these (unit) vectors on their own rows.

3. Scan Part 1 of V-region

   Establish row counts: Search for entries in rows where the H-value is negative and subtract 1 from this H-value. If there are no such rows, the vector cannot be introduced. Write message to declare MATRIX SINGULAR and assign complementary unit vector to basis. Replace vector by the bottom entry in Part 1 of V-region.

If there is only one such row, the vector will pivot below the bump in this row, if the pivot is larger than ZTOLPV.

    replace vector by the bottom entry in Part 1 of V-region
    put sequence number into Part 3 of V-region
    put pivot into Part 3 of M-region
    put 0 into H-region

Otherwise go on to next vector.

4. Repeat following step until no more vectors can be removed from bump.

Scan Part 1 of V-region

If a vector has an element in a row where the H-value is -2 then this is the only column with an element in that row and the vector will pivot above the bump.

    Write ETA to pivot in this row.
    Add 1 to negative entries in rows where this vector has a coefficient.
    Put sequence number into H-region.
    Replace vector by bottom entry in Part 1 of V-region.

Otherwise test the count of entries in any rows where the H-value is negative.

If there are no such rows, the vector cannot be introduced. Write message to declare MATRIX SINGULAR and assign complementary unit vector to the basis. Replace vector by the bottom entry in Part 1 of V-region.

If there is only one such row, the vector will pivot below the bump in this row, if the pivot is larger than ZTOLPV

Replace vector by the bottom entry in Part 1 of V-region.

Put sequence number into Part 3 of V-region.

Put pivot into Part 3 of M-region.

Put 0 into H-region.

If there is more than one such row, calculate the Merit to be put into the M-region as the sum of (row count -1) over all rows with HREG(I) < 0.

Sort rows in Part 1 of V-region into acsending merit order.

5. Write out below bump etas for nonunit vectors.

6. Take vectors in Part 1 of V-region in turn.

Unpack entries in vector into Y(·)

Do FTRAN on vector.

Set pivot tolerance ZTOLY to ZTOLPV

Select pivotal row as the one with coefficient above ZTOLY in magnitude that has the smallest row count. If selected pivot has absolute value less than ZTREL*(maximum eligible Y(I)) reset ZTOLY to this value and repeat row selection.

Write L and U etas pivoting on this row.

Increase row counts in all rows where ETA nonzero value and H-region is negative by (pivotal row count -2), i.e. increase H-region by (pivotal H-value + 3).

If no rows have a coefficient above ZTOLPV in magnitude then this vector cannot be introduced. Write message to declare MATRIX SINGULAR and assign complementary unit vector to basis

7. Merge L and U etas (calling SHFTE)
8. If singularities were detected, print message and exit
9. Update solution, i.e. compute X(·)
10. Print statistics
11. Check relative error in solution (using CHSOL) and if the value is excessive repeat INVERT with ZTREL increased to 0-5.

Messages

MATRIX SINGULAR-VECTOR 'column name' REMOVED

INVERT has detected a singularity. See below.

INVERT FAILS. WILL ATTEMPT TO RECOVR

One or more singularities were detected by INVERT. A call to RECOVR is triggered.

INSUFFICIENT ETA SPACE TO INVERT-RUN ABANDONED

INCREASE ETA SPACE AND NEMAX

The arrays IE(·) and E(·) in the program must be increased from their current dimension (standard value is NEMAX = 8000).

Subroutines Called

WRETA

FTRAN

SHIFTR

SHFTE

CHSOL

ITEROP

Prints the iteration log (see "Users Guide to LCPL").

Formal Declaration

```
SUBROUTINE ITEROP (IPAR)
```

Parameter: IPAR(Input). When IPAR=0 print headings for iteration log. When IPAR ≠ 0 print a line of the iteration log.

Information Printed

ITCOUNT	=	iteration
STATUS	=	A almost complementary C complementary U unbounded almost complementary ray
DUMMY Z	=	value of the "dummy variable"
VECIN	=	name of column entering basis
VECOUT	=	name of column leaving basis
NETA	=	number of etas in the eta file
NELEM	=	number of elements in the eta file.

LEMKE

Main algorithmic routine controlling the steps of the algorithm, inversion and error recovery (if necessary).

Formal Declaration

```
SUBROUTINE LEMKE
```

Method

0/ It is assumed that LEMKE is called with a feasible, almost complementary basis found in INPUT.

1/ INVERT is called. If the basis is singular a call to RECOVR is triggered. When (or if) a new nonsingular almost complementary basis has been found in RECOVR the INVERT routine is called again.

2/ After INVERT headings are printed out for the iteration log (by a call to ITEROP(0)).

3/ The row IROWZ on which "DUMMY Z" pivot is found.

4/ Test the variable NONC which has just left the basis to see if it is a w variable (NONC $\leq$ NROW), in which case JCOLP = NROW + NONC is to enter the basis, or a z variable (NONC > NROW) in which case JCOLP = NONC - NROW is to enter basis.

5/ Column JCOLP is UNPACKed and FTRANed.

6/ CHUZR (IFPV)is called. If IFPV=1 proceed to step 7/.
If IFPV $\leq$ 0 go to step 1/and INVERT unless inversion has just been performed. If ITSINV=0 and IFPV=-1 the solution is infeasible--go to RECOVR and step 1/. If ITSINV=0 and IFPV=0 the pivot row IROWP chosen in CHUZR must be accepted.

7/ Bring column JCOLP into the basis, pivoting on row IROWP and ejecting the column JH(IROWP). Redefine NONC.

8/ Update X($\cdot$) and print a line of the iteration log.

9/ If "DUMMY Z" (column NCOL) has left the basis or its value is less than ZTOLZE go to 13/ and declare a complementary solution

10/ Write a new eta

11/ If it is time to do, write out the basis, call CHSOL or call INVERT (i.e. go to 1/).

12/ If ITCNT is $\geq$ ITRLIM terminate and go to 13/

13/ Write the final line of the iteration log giving terminal status, the final basis, and return to the main program.

<u>Message</u>

ITERATION LIMIT EXCEEDED

<u>Subroutines Called</u>

RECOVR	UNPACK	UPBETA	BASOUT
INVERT	FTRAN	WRETA	
ITEROP	CHUZR	CHSOL	

MC12A

Harwell subroutine for scaling a square matrix using the method of Curtis and Reid.

Formal Declaration

SUBROUTINE MC12A (A, IND, IP, N, NP, DIAG, RES, IS)

The parameters and the methods are fully explained and described in

A.R. Curtis and J.K. Reid, "Fortran Subroutines for the Solution of Sparse Sets of Linear Equations," Report AERE-R 6844, Theoretical Physics Division, AERE, Harwell, England (1971).

Note

Very minor changes have been made so that nested DO-loops do not end on the same statement, thus achieving WATFIV compatibility.

OUTPUT

Write the solution found by LEMKE to FORTRAN logical unit KOUTP.

Formal Declaration

SUBROUTINE OUTPUT

Method

0/ Write the problem name at the top of a new page.

1/ If IØUT=0 go to 6/ (Standard)

2/ Write a heading

3/ Sort the basic variable values X(·) into matrix order using the sequence numbers in JH(·)

4/ For I = 1, NROW write the name and value of a basic variable (in sorted order), the row name and the value of the original right-hand side element B(I).

5/ Go to 9/.

6/ Write a heading

7/ For I=1, NROW write the name and value of the w_i and z_i variables

8/ If "DUMMY Z" is basic write its name and value under the Z variables.

9/ Compute and write the value of the quadratic objective value if NQUAD has been specified.

Note If IFSCAL=1, all the solution values must be descaled before pivoting and before computing the quadratic objective function (see SCALE for details of scaling factors).

RECOVR

Construct a new nonsingular almost complementary feasible basis (if this is possible) from the old basis using as many of its structured columns as possible.

Formal Declaration

SUBROUTINE RECOVR

Method

1/ Remove "DUMMY Z" from the basis and replace it by NONC to give a complementary basis

2/ Check for correct number of basic vectors and reset JH(·) to include them. Count number of structural columns (columns of M) in basis, denoted NSTB.

3/ Attempt to invert basis. If singularity was found repeat until for a maximum of NSTB tries until a nonsingular basis is found (INVERT automatically ejects dependent structural columns and replaces them by their complementary unit column)

4/ Examine the basic solution in array X(·). Whenever a negative X(I) is found subtract the vector IV=JH(I) from region Y(·) (initialized to zero). Record row IROWZ containing the most negative entry

5/ If solution is nonnegative (i.e. IROWZ=0) declare termination since the solution has been constructed to be complementary.

6/ If IROWZ is nonzero, pack the column in IA(·) and A(·). Write a message that this has been done.

7/ Define NONC as JH(IROWZ) and change JH(·) and KINBAS(·) so that the new basis has "DUMMY Z" (i.e., column NCOL) basis in place of NONC. Return. (INVERT will be called again on return for the new almost complementary feasible basis.)

Messages

FATAL ERROR-'number' BASIC VECTORS

The problem does not have NROW basic vectors. Probably due to corrupted program or split vectors.

COMPLEMENTARY BASIS FOUND

RECOVR has found a new complementary basis.

FEAS COMP BASIS ARRIVED AT IN RECOVR

RECOVR has (fortuitously) found the complementary basis it has constructed to be feasible. Problem solved.

NEW DUMMY COLUMN CONSTRUCTED

RECOVR has constructed a new dummy column and determined a pivot row which will give a feasible almost complementary solution. Returns to normal algorithm.

Subroutines Called

INVERT

SCALE

Sets up arrays and parameters for calling the Harwell scaling subroutine MC12A and then applies the scaling factors returned by MC12A to the matrix (M) and the right-hand side stored in B(·).

Formal Declaration

SUBROUTINE SCALE

Special Techniques

Scaling factors are applied by adding integers to the floating point exponents of A(·) (single precision) and B(·) (double precision). This is accomplished by means of equivalences. Single and a double precision variables AVAL and BVAL are defined which are equivalenced to LOGICAL*1 variables IU and IV. IU and IV are thus the exponents of AVAL and BVAL with the standard bias of 64 (decimal) added. Any matrix element A(I), or right-hand element B(I), is thus stored in AVAL or BVAL and the exponent is extracted or modified by examining or changing IU or IV. The modified AVAL or BVAL then overwrites A(I) or B(I) if a change has been made.

In order to perform arithmetic with the IU and IV a LOGICAL*1 array IW(·) of dimension 4 is defined and equivalenced with an INTEGER*4 variable IVAL. IU and IV are thus placed in, or extracted from, the rightmost byte of IVAL by passing them through IW(4).

Parameters Passed

MC12A requires as parameters the arrays A(·), IA(·) and ISC(.,2) which are in blank common for all other subroutines. An array IP(.,2) with first dimension NRMAX+1 is also passed. This array is equivalenced to IE(·). IP(I,1) contains IA(NRØW+I) for I=1, NROW+1, the pointers to the structural columns (the M matrix), while IP(I,2) contains only the integers 1 through NROW+1. A REAL*4 array WSPACE(·) of dimension 4*NRMAX, equivalenced to Y(·) is also passed. IP(.,2) and WSPACE(·) must be redimensioned if NRMAX is changed.

Method

1/ First the sum of squares of exponents of the M matrix is computed and printed.

2/ The array IP(.,2) is set up and MC12A is called.

3/ If MC12A detects an empty row or column scaling is abandoned.

4/ After MC12A has returned the row and column scales for M in the array ISC(.,2) the structural column in A(·) and the right-hand side B(·) are scaled by modifying their exponents.

5/ The new sum of squares of the exponents of the one matrix is computed and printed.

Messages

MATRIX SCALING CALLED FOR

IFSCAL was set equal to a nonzero value and M will be scaled.

INITIAL SUM OF SQUARES OF EXPONENTS = 'number'

Initial status of matrix before scaling by exponent.

NEW SUM OF SQUARES OF EXPONENTS = 'number'.

Status of matrix after scaling. All scaling is done on exponents only, and hence scaling factors are powers of 16. If the matrix was well scaled to begin with, the new sum of squares may be larger than the old.

SINGULAR M MATRIX-NO SCALING PERFORMED

The scaling routine detected a zero row or column. The run will continue but normal termination is unlikely.

SUBROUTINE CALLED

MC12A

SHFTE

Shifts the etas corresponding to the upper triangular factor (U) produced by INVERT when it carries out an LU decomposition, so that they are contiguous with the etas for the L factor. The unified eta file is then ready for standard product form application and update.

Formal Declaration

```
SUBROUTINE SHFTE
```

Method

The U (backward) etas were built up in reverse order from the bottom of the LE(·), IE(·) and E(·) regions starting at NTMAX+1, NEMAX and NEMAX, respectively. The number of U etas (NUETA) and eta elements (NUELEM) is known. Given NLETA and NLELEM the U etas are shifted to follow on directly from the L etas.

SHIFTR

Shifts the contents of one region to another (completely over-writing the first NROW elements).

Formal Declaration

SUBROUTINE SHIFTR(IØLD,INEW)

Parameters:	IØLD (Input):	Region number to be copied (shifted)
	INEW (Input):	Region number into which region IØLD is to be shifted (copied).

Notes

1/ The regions B(.), X(.), Y(.), YTEMP(.) are numbered 1, 2, 3 and 4 for the purposes of this routine and the parameters IOLD, INEW must take values 1, 2, 3, or 4.

2/ The array BARRAY(·), equivalenced to B(·) must be given the dimension 4*NRMAX if NRMAX is charged.

UNPACK

Unpacks a packed vector in the coefficient matrix (A) into the work region Y(·).

Formal Declaration

SUBROUTINE UNPACK (IV)

Parameters: IV(Input)--the sequence number (column number) of the column to be unpacked

Method

The vector is unpacked exactly as described in the "Matrix Storage" section.

UPBETA

Updates the solution vector X(·) when Y(IROWP) is pivoted on.

Formal Declaration

SUBROUTINE UPBETA

Method

Standard pivoting procedure

1/ Compute new DP = X(IRØWP)/Y(IROWP)

2/ Update X(I) = X(I) - Y(I)*DP for I = 1,NROW

3/ Replace X(IROWP) = DP

WRETA

Write a new eta to the end of the eta file using the vector in Y(·) and IROWP.

Formal Declaration

SUBROUTINE WRETA

Method

1/ Increase NELEM by 1 and write

IE(NELEM) = IROWP

E(NELEM) = Y(IROWP)

(see section "Transformation Processing")

2/ Zero out Y(IROWP)

3/ Pick up all elements in Y(·) with absolute value greater than ZTETA and pack sequentially in IE(·) and E(·), increasing NELEM at each step

4/ Increase NETA by 1 and define LE(NETA+1) = NELEM+1

XCOLS

Constructs extra columns to make up a square matrix when NQUAD has been specified and only NQUAD (< NROW) columns were supplied. The matrix M is assumed to correspond to a quadratic program and be of the form

$$\begin{bmatrix} D & -A^T \\ A & 0 \end{bmatrix} .$$

XCOLS picks up the last NROW-NQUAD rows of the matrix supplied and constructs $\begin{bmatrix} -A^T \\ 0 \end{bmatrix}$, which columns are added on to the assumed $\begin{bmatrix} D \\ A \end{bmatrix}$ to give the full M. No empty rows of A are allowed.

Formal Declaration

SUBROUTINE XCOLS

Method

Search rows NQUAD+1 through NROW in sequence for nonzeros. Store their negatives as new columns.

Messages

NUMBER OF ELEMENT EXCEEDS NAMAX-ABORT

The new elements exceed storage capacity.

TRANSPOSE COLUMNS ADDED TO MATRIX

Successful completion

EMPTY RØW 'row no.' ENCOUNTERED-ABANDONING PROBLEM

An empty row (number 'row no.') has been encountered in trying to construct $-A^T$. Empty columns are not allowed. Fatal error.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 SOL-76-25	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 PROGRAMMERS GUIDE TO LCPL: A Program for Solving Linear Complementarity Problems by Lemke's Method.		5. TYPE OF REPORT & PERIOD COVERED 9 Technical Report.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 J. A./Tomlin		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-75-C-0865, DAAG-29-74-C-0034
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Operations Research Program Code 434 Office of Naval Research Arlington, Virginia 22217 Mathematics Division U.S. Army Research Office Box CM, Duke Station Durham, North Carolina 27706		12. REPORT DATE 11 October 1976
		13. NUMBER OF PAGES 53 12 58p.
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Linear Complementarity
Mathematical Programming
Quadratic Programming
Sparse Matrices

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report gives programmers information and documentation for LCPL--an efficient robust program for solving Linear Complementarity Problems by Lemke's Method.

408 765 ✓

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)